MANUAL DE UTILIZAÇÃO DO II2 – INSTANT INTEGRATOR 2

versão 3.36

(http://www.ii2.com.br)

Sumário

Anotações:

Sobre o II2

O Instant Integrator 2 é uma ferramenta que foi desenvolvida com o intuito de facilitar e automatizar a integração entre banco de dados. Sua função principal é a criação de TRIGGERS em bancos de dados que farão o trabalho de replicar as ações ocorridas num banco de dados de ORIGEM em um outro banco de dados (DESTINO).

Através de uma interface simples e amigável é possível fazer as ligações entre as tabelas e campos dos bancos de dados envolvidos. É possível também trabalhar com diversos recursos avançados que possibilitam atender a todas as necessidades que uma integração complexa entre bancos de dados de fornecedores diferentes possa apresentar.

Além de criar as TRIGGERS e aplica-las ao banco, a ferramenta possui um mecanismo de monitoramento que identifica quando um OBJETO no banco de dados estiver inválido e recriá-lo automaticamente de acordo com as novas definições de estrutura dos bancos de dados, impedindo assim que a integração deixe de funcionar, impedindo que os dados entre os bancos se tornem inconsistentes. Durante o monitoramento, caso o processo de recriação de um OBJETO inválido não resolva o problema, ou seja, o OBJETO continue inválido, a ferramenta automaticamente envia um e-mail ao responsável pelo banco de dados (parametrizável) informando que a integração poderá ter algum problema, além de gravar esta ocorrência em um arquivo de LOG.

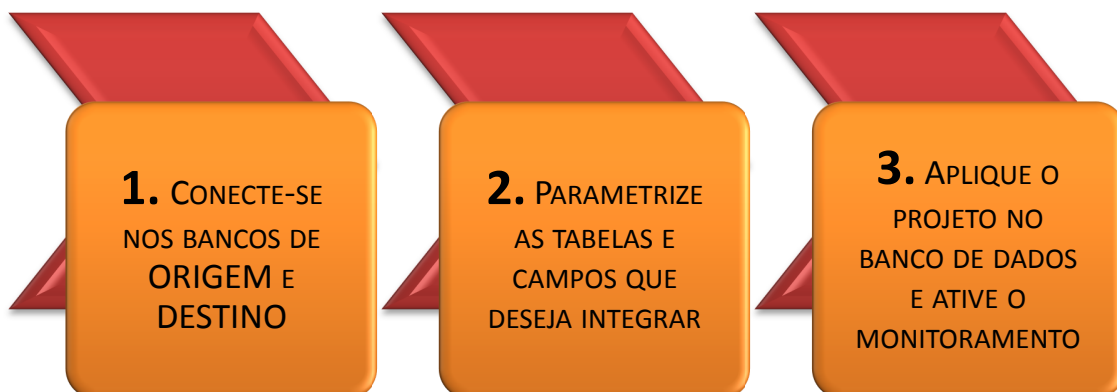
Porém sua utilização não se restringe apenas a criação das TRIGGERS e monitoramento do banco de dados, mas um dos pontos de maior utilidade desta ferramenta é a manutenção que ocorre nestes OBJETOS durante a atualização dos sistemas ao qual foi arquitetada a integração, quer seja na ORIGEM dos dados ou no DESTINO. Visto que, a criação das TRIGGERS se dá através da conexão com os bancos de dados e a parametrização das tabelas e campos que se deseja integrar, a ferramenta sempre efetuará uma nova leitura da estrutura atual dos bancos de dados e ao recriar as TRIGGERS utiliza-se das definições atuais quer seja de campos, e seus novos tipos e tamanho e atributos.

✍ Anotações:

- Como a integração dos bancos de dados criada através da ferramenta é totalmente baseada em TRIGGERS é extremamente recomendado que durante o processo de **atualização dos sistemas**, os projetos de integração sejam desativados e ativados novamente somente após o término e validação da atualização efetuada, com isso, evita-se que algum processamento em massa que venha a ser executado durante a atualização corrompa os dados na tabela de DESTINO, além de manter a TRIGGER atualizada com a nova estrutura dos bancos de dados.

Após a criação de projetos de integração entre bancos de dados, é possível a utilização deste mesmo projeto em outros bancos de dados, diferentes do original ou não (ORACLE, MS SQLServer, etc) pois a linguagem e comandos utilizados na criação dos OBJETOS se adequará automaticamente ao banco de dados de sua nova conexão.

E ainda, a ferramenta possibilita após a criação de um projeto, a utilização de comandos customizados necessário na utilização de projetos em ambientes, servidores, bancos de dados diferentes que necessitem de novas funcionalidades das criadas automaticamente pela ferramenta.

- Um projeto de integração pode ser criado de maneira fácil e rápida em apenas 3 passos:

Anotações:

Conceitos básicos de Bancos de Dados Relacionais

Bancos de dados relacionais são baseados em tabelas e os relacionamentos entre elas. Estas tabelas são sempre bidimensionais, onde no eixo "x" têm-se as colunas, e no eixo "y" têm-se os registros. Um registro é formado por dados, ou valores, para as respectivas colunas.

- Desta forma, considerando-se uma tabela com três colunas, tais como "Nome", "Telefone" e "e-mail" um registro teria três dados, ou valores, um para cada coluna.
  Por exemplo, (Paulo, 11 12345678, paulo@meuemail.com) seria um registro, enquanto que (Pedro, 11 98765432, pedro@myserver.com) seria outro registro.

Os bancos de dados relacionais provêm uma maneira única de qualquer aplicação acessar seus dados. Fazem isso através de um padrão de armazenamento próprio e de um SGBD.

- SGBD (Sistema Gerenciador de Banco de Dados) cria uma interface entre o programa e o banco de dados em si e é ele quem grava ou lê os dados do banco. O programa não precisa conhecer as complexas estruturas de dados e índices usados no arquivo de banco de dados para acessá-los. Basta saber como se comunicar com o SGBD que este fará todo o processo.

O SGBD assume a responsabilidade de ler, gravar, garantir ou revogar acesso, indexar e manter o banco de dados integro. Para que o programa conheça e saiba se comunicar com o SGBD é necessária ainda outra camada, que são os drivers de bancos de dados, também chamados de bibliotecas clients. A comunicação é feita através da Structured Query Language ou SQL.

O SQL e o modelo relacional fornecem um padrão para que absolutamente qualquer conjunto de dados, de qualquer tipo, possa ser representado numa estrutura de tabelas, colunas, registros e relacionamentos.

Qualquer banco de dados relacional pode ser facilmente representado graficamente em um diagrama chamado diagrama de entidade-relacionamento (DER) ou modelo de entidade-relacionamento (MER). Esse diagrama é um dos melhores instrumentos para representarmos o negócio ao qual o banco de dados em questão foi proposto.

Anotações:

O ideal e mais comum é partir de um diagrama desse tipo para desenvolver todo o banco de dados e o sistema, e criar desta forma um sistema totalmente orientado aos dados.

Drivers

Para que um programa feito em uma linguagem qualquer se comunique com um sistema gerenciador de banco de dados qualquer é necessário um driver. Driver nesse contexto significa um programa ou DLL que é capaz estabelecer uma comunicação entre o programa desenvolvido e o SGBD.

Os SGBDs abrem portas TCP, UDP ou ambas para que os programas possam se comunicar com eles. Todas as transferências de dados ocorrem por essas portas. O que o programa envia para o SGBD são comandos SQL de inserção, atualização, consultas ou ainda um conjunto de comandos para serem armazenados no banco de dados tais como TRIGGERS e STORED PROCEDURES, e o que o SGBD responde pode ser a quantidade de registros alterados ou o conjunto de registros em si.

Para cada tipo de banco de dados o II2 utiliza e disponibiliza alguns drivers (DLLS) para a comunicação com o mesmo.

TABELAS

Uma tabela ou entidade é uma representação abstrata de um conjunto de dados bidimensionais definidos através de uma estrutura de colunas (campos ou eixo “x”) e linhas (registros ou eixo “y”).

Um campo é a definição mais precisa de um dado (nome, endereço, e-mail etc). O SGBD tem a capacidade de impor uma camada de regras para o correto preenchimento de valores a um campo. São as regras de validação ou consistência (TRIGGERS e/ou CHECKS).

Após a definição dos campos de uma tabela, ou da estruturação da entidade, a mesma está apta a receber os dados. Os dados são cadastrados em linhas nas tabelas onde cada uma das linhas representa um conjunto único de dados. As linhas também são denominadas de registros.

Anotações:

CHAVES

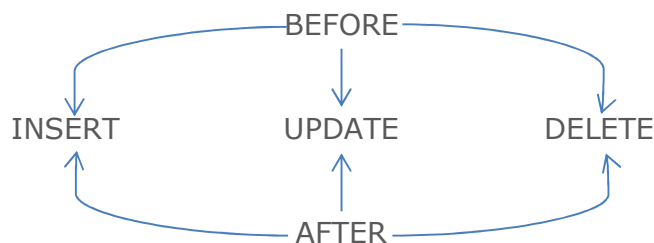
As tabelas de um banco de dados relacional podem conter chaves que nada mais são que identificadores de registros. As chaves de uma tabela são responsáveis por manter a integridade dos dados de uma tabela e referenciar os relacionamentos entre estas entidades.
Podemos dividir as chaves da seguinte forma:

- **Chave primária (Primary Key):**
  A chave primária é o identificador de registros de uma tabela. Elas contem um valor único e obrigatório. Podem ser formadas por um ou mais campos (neste caso chama-se chave primária composta).
  Imagine a seguinte situação: Um cadastro de clientes onde seja necessário fazer buscas e filtro dentro do banco de dados, como por exemplo, listar os clientes cadastrados numa determinada data ou os clientes com status de ativo. Para isso será necessário identificar os clientes por um campo que tenha um valor único, ou seja, um valor que nunca mais se repita dentro da tabela. Para esta finalidade, o CPF seria um ótimo campo, pois todos os clientes devem ter CPF no ato do cadastro (obrigatório) e todos os CPFs são únicos (não existem CPFs iguais). Uma ótima técnica utilizada é a criação de campos próprios no banco de dados para representarem os campos primários tais como ID, Código, Matrícula, etc.
- **Chave estrangeira (Foreign Key):**
  A chave estrangeira nada mais é que a referência à outra tabela (relacionamento). Consideremos que para cada cliente tenhamos um histórico de faturamentos, logo os faturamentos devem ser identificados como pertencentes a este cliente. Para isso deve-se criar uma referência ao cliente, ou seja, definir que o faturamento “X” pertence ao cliente “Y”. Para isto precisamos fazer um clone da chave primária da tabela “origem” (cliente neste caso) para a tabela “destino” (faturamento).

Anotações:

TRIGGERS

Uma TRIGGER é um tipo especial de procedimento codificado armazenado (conjunto de comandos SQL) no próprio banco de dados e ligado a uma tabela específica, que é executada sempre que houver uma tentativa de modificação de dados, ou seja, quando houver uma tentativa de inserir (INSERT), atualizar (UPDATE) ou excluir (DELETE) os dados desta tabela em que a TRIGGER estiver ativa e parametrizada para essa ação específica, ela será executada automaticamente, não podendo nunca ser ignorada.

As TRIGGERS podem ser parametrizadas para serem executas antes (BEFORE) ou depois (AFTER) da ação de modificação de dados da tabela ocorrer e diferente das PROCEDURES (que veremos logo a seguir), as TRIGGERS não podem ser executadas através de chamadas diretas por código e nem aceitam parâmetros. Sendo assim, temos seis situações onde as TRIGGERS podem ser parametrizadas, conforme demonstrado abaixo:

A chamada da TRIGGER com as instruções que serão executadas por ela fazem parte da mesma transação (TRANSACTION) a qual a acionou podendo desta forma ser revertida a qualquer momento através do comando de banco de dados ROLLBACK TRANSACTION.

Para que uma TRIGGER seja disparada, o usuário de conexão com o banco de dados obrigatoriamente deverá ter permissão de acessar tanto a tabela quanto a TRIGGER.

Usos e aplicabilidades das TRIGGERS:

- Impor integridade de dados de forma mais complexa do que uma restrição do tipo CHECK;
- Definir mensagens de erros personalizadas;
- Comparar informações e manter a consistência dos dados – posterior e/ou anterior à execução uma instrução (UPDATE, por exemplo);
- **Manter dados sincronizados entre bancos de dados distintos, tratando informações e garantindo a integridade das mesmas.**

As TRIGGERS são utilizadas com eficiência para impor e manter integridade referencial de dados, e não para retornar resultados de consultas. A principal vantagem é que elas podem conter uma lógica de processamento complexa.

Podemos utilizar TRIGGERS para atualizações e exclusões em cascata através de tabelas relacionadas em um banco de dados ou em bancos de dados distintos, impor integridades mais complexas do que uma restrição *CHECK* de um banco de dados, definir mensagens de erro personalizadas e fazer comparações dos momentos anteriores e posteriores a uma transação.

- ❖ Podemos definir integridade referencial através do uso das restrições *FOREIGN KEY* e *REFERENCE.* Nestes casos as TRIGGERS fazem bem o trabalho de checagem de violações e garantem que haja coerência de acordo com a regra de negócios.

Usando como exemplo a exclusão de um cliente, certamente teríamos que excluir também todo o histórico de movimentações. O uso de TRIGGERS neste caso garantiria que o processo fosse executado como um todo e não apenas uma parte desta transação.

Anotações:

STORED PROCEDURES

A definição mais comum do termo diz que: *Stored Procedure*, ou no português, Procedimento Armazenado, é um conjunto de comandos *SQL* que juntos formam uma rotina ou sub-rotina. Própria dos *databases* relacionais, a ideia é que as *Stored Procedures* fiquem armazenadas junto com os dados, dentro do mesmo *database*.

Em termos práticos isso significa que ao executarmos essas rotinas, temos dois benefícios significativos: a execução dos comandos é mais rápida já que eles não precisaram ser transferidos pela rede, a única coisa que fazemos é indicar qual *Stored Procedure* queremos executar. Seu uso torna a aplicação mais segura, já que há uma chance muito menor dos comandos *SQL* serem intencionalmente modificados no caminho que percorreriam entre o Cliente e o Servidor em casos onde não se utiliza *Stored Procedures*.

Elas são criadas como recursos adicionais dos bancos de dados relacionais que vão muito além da capacidade nata de armazenamento de dados, tornando os Databases uma ferramenta de armazenamento de dados muito mais robusta.

Conceito de Integração de Dados

Integração de Dados ou Integração de Bancos de Dados consiste em manter informações sincronizadas entre bancos de dados, sendo estes atualizados automaticamente, mantendo integra as informações e consistente as entidades relacionais evitando desta forma, o retrabalho (ou redigitação de dados) e garantindo a origem das informações devido à automação da mesma.

Uma integração de dados pode se dar de diversas formas:

- Consumo de WebService
- Envio de arquivos TXT
- Robôs (programas) criados especialmente para esta finalidade
- TRIGGERS de banco de dados

Quando tratamos de integração entre banco de dados de fornecedores distintos, integrações do tipo WebService, TXT ou Robôs dependeremos necessariamente das aplicações estarem preparadas para receberem este tipo de integração, ou do desenvolvimento de rotinas proprietárias para esta finalidade, o que, em muitos casos o custo é alto e acaba por inviabilizar o

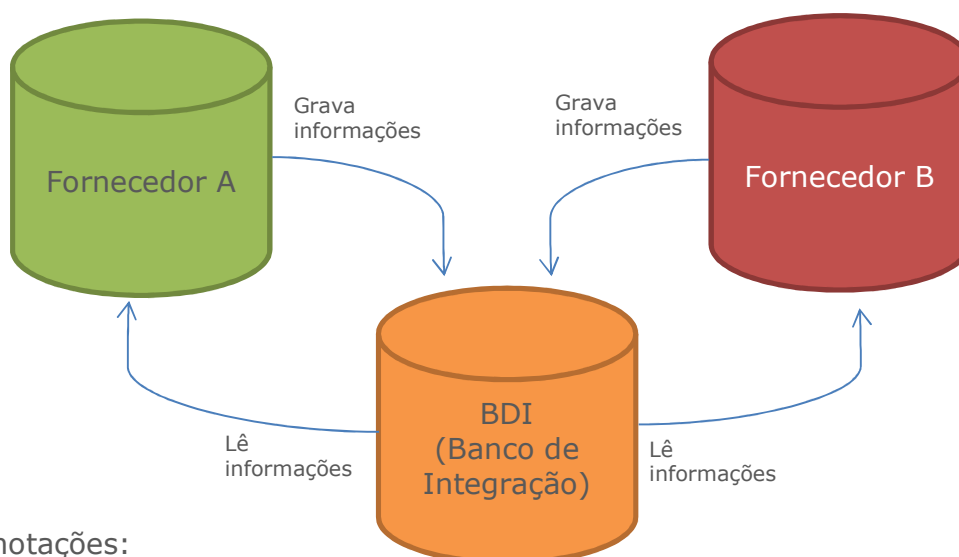
Anotações:

projeto, além do alto custo de manutenção para este tipo de integração que a cada atualização de versão de qualquer das aplicações (ORIGEM ou DESTINO).

Considerando o exposto, a integração via TRIGGER é uma saída para a não alteração de aplicações, pois a regra de envio e recepção de informações ficam registradas no próprio banco de dados, sendo que um profissional com conhecimento na estrutura de tabelas dos bancos de dados (ORIGEM e DESTINO) pode efetuar esta programação para garantir o bom funcionamento da integração, todavia, ainda carece de manutenção toda vez que ocorrer a atualização de uma das aplicações tornado custoso manter uma integração de dados ativa.

Ainda, devemos considerar um ponto de muita importância quando falamos de integração entre bancos de dados distintos, que é a permissão de acesso de gravação entre os bancos de dados de fornecedores de soluções diferentes, o que, na prática, nunca ocorre devido o cuidado em perderem-se dados ou tornar informações inconsistentes dentro de uma estrutura complexa de bancos de dados.

Para isso, utilizaremos alguns conceitos e boas praticas no desenvolvimento de integração complexas entre bancos de dados a seguir:

1. Para efetuarmos a integração entre dois ou mais bancos de dados distintos ou de fornecedores diferentes de soluções que compartilham cadastros semelhantes de dados, boa prática é a utilização de um BDI (Banco de Dados de Integração), onde, ambos fornecedores compartilhem informações, gravando e lendo dados deste, não sendo necessário a liberação de acesso a base de dados para um outro fornecedor, desta forma, a integração ficaria funcionando conforme modelo abaixo:

Anotações:

- É importante que seja definido um único ponto de ORIGEM para cada uma das tabelas ou entidades que forem consideradas um projeto de integração de dados para evitar problemas com sincronismo e perda de dados nas transações.
  Considerando um exemplo de cadastro de clientes e dois sistemas de fornecedores de soluções distintas um para controle de contratos e processos operacionais, e outro para controle de faturamento e contabilidade. O ideal é que a ORIGEM do cadastro seja definido em uma das aplicações e a outra seja utilizada apenas para preencher informações complementares.

2. A manutenção e monitoramento das TRIGGERS após um processo de atualização das aplicações (quer seja da aplicação de ORIGEM dos dados ou da aplicação de DESTINO dos dados) é de extrema importância neste tipo de integração para garantir que a integridade e sincronismo dos dados ocorram sem problemas garantindo o bom funcionamento de ambas as aplicações.
   - É importante atentar-se que a manutenção em um banco de dados quer seja por motivo da atualização de uma aplicação ou não pode invalidar um OBJETO (TRIGGERS, STORED PROCEDURES etc) devido a referencia a alguma entidade, campo, índice que tenha deixado de existir ou tenha sido alterado, por este motivo é essencial o monitoramento do status destes OBJETOS para que quando ocorra um caso destes uma ação corretiva seja tomada de imediato.

Para garantir estes pontos apresentaremos a seguir a melhor maneira de utilização da ferramenta II2 e como tirar o melhor proveito de todas as funcionalidades que ela tem a oferecer.

Anotações:

INICIANDO A UTILIZAÇÃO DO II2

Estrutura física e instalação do II2

O II2 – Instant Integrator 2 não necessita de processos de instalação automatizados, sendo necessário apenas copiar os arquivos que compõe a ferramenta em uma pasta em seu HD no servidor que a ferramenta já irá funcionar normalmente. Os arquivos da ferramenta podem ser adquiridos diretamente no site oficial (http://www.ii2.com.br) e de lá também é possível obter a atualização, visualizar os releases e a ultima documentação com as novas funcionalidades disponibilizadas na ferramenta.

Estrutura de arquivos na pasta do II2:

Arquivo	Descrição
II2.EXE	Arquivo principal do II2 o qual deve ser utilizado para entrar na ferramenta
II2PROP.GAM	Arquivo de proprietária da ferramenta que contem os dados de registro e licença da ferramenta. Este arquivo não pode sofrer nenhuma alteração, pois se ocorrer a ferramenta para de funcionar
DBXEXPODA30.DLL	Arquivo de biblioteca fornecido para conexão com o ORACLE
DBXEXPODA40.DLL	Arquivo de biblioteca fornecido para conexão com o ORACLE
DBXEXPSDA30.DLL	Arquivo de biblioteca fornecido para conexão com o MSSQL Server
DBXEXPSDA40.DLL	Arquivo de biblioteca fornecido para conexão com o MSSQL Server
II2_USERLIBRARY.INI	Arquivo que contém as palavras reservadas criadas pelo usuário na ABA UL do II2
II2_SERVERS.INI	Arquivo que contém o registro dos servidores que podem ser cadastrados pelos usuários para facilitar os testes e troca de servidores para utilização de um mesmo projeto em ambientes distintos

Anotações:

II2_TOOLS.INI	Arquivo que receberá as configurações da ferramenta independente dos projetos. Configurações tais como aparência da ferramenta.
II2_[PROJETO].INI	Arquivo que contém os dados de conexão com os servidores de ORIGEM e DESTINO do projeto criado além das informações pertinentes ao projeto. [PROJETO] é o nome do projeto que é criado na ABA de conexão do II2 para identificação do mesmo quando se utiliza diversos projetos de integração em mais de um servidor
II2_[PROJETO]_TABLES.INI	Arquivo que contém as parametrizações das integrações guardando informações de tabelas ORIGEM e DESTINO, campos, lista DE/PARA, comandos avançados entre outros.
II2_[PROJETO]_OBJECTS.INI	Arquivo que contém o registro dos OBJETOS AVULSOS (procedures) que devem ser criados antes da criação das TRIGGERs
II2_[PROJETO]_CUSTOM.INI	Arquivos que contém as informações customizadas dos projetos que são inseridos pelos usuários para personalização das parametrizações em ambientes distintos ou situações que diferem do ambiente em que um mesmo projeto foi criado. Estas informações são inseridas no II2 nos campos identificados como (CUSTOM) e em azul sendo que fazem referencia exclusiva a um determinado ambiente não precisando ser copiado para outros servidores quando se deseja replicar o mesmo projeto em diversos servidores

Anotações:

Conectando-se aos bancos de dados

Para iniciar um projeto, os primeiros passos devem ser a parametrização da conexão dos bancos de dados de ORIGEM e DESTINO, ao ser executado o II2 a primeira tela que é apresentada é a tela de conexão conforme demonstrado abaixo:

Entendemos esta tela através de cada campo:

- **[Nome da Integração]**: O primeiro passo é iniciar um projeto novo, criando um nome para ele neste campo. Atenção que todos os arquivos INI referente a este projeto serão criados com este nome;
- **[Sigla da Integração]**: Este campo deve ser preenchido com uma sigla que identifique o projeto e o diferencie dos demais projetos de

Anotações:

integração. O conteúdo deste campo fará parte do nome das TRIGGERs que serão criadas automaticamente pela ferramenta. Com ele será possível a criação de duas ou mais TRIGGERs em uma mesma tabela, porém em projetos diferentes, possibilitando assim a integração de uma ORIGEM com dois ou mais DESTINOS distinto;

- **[Comentários da Integração]**: Utilize este campo para fazer comentários referentes ao projeto de integração, este comentário fará parte da documentação gerada pela ferramenta;
- **[Comentários da Integração (Custom)]**: Os comentários inseridos neste campo serão gravados num arquivo diferente e serve para documentar as diferenças de integração de um mesmo projeto mas em ambientes diferentes. Estas informações também farão parte da documentação gerada pela ferramenta;
- Informações pertinentes a conexão com os bancos de dados (ORIGEM e DESTINO):
    - **[Driver Name]**: Selecione neste campo o tipo de banco de dados que se efetuará a conexão (ORACLE, MSSQL Server etc);
    - **[Host Name]**: (Somente para conexão com MSSQL) Informe neste campo o nome do servidor e instancia do SQL Server onde se encontra o banco de dados;
    - **[DataBase]** (MSSQL) / **[TNSNAMES]** (ORACLE): Informe neste campo o nome do database para conexão com MSSQL e/ou o nome da conexão do ORACLE registrada no TNSNAMES.ORA;
    - **[USER NAME]**: Nome do usuário com acesso SYSDBA no banco de dados. É necessário permissão SYSDBA nos bancos de dados ORIGEM e DESTINO para que a ferramenta possa buscar as informações das estruturas de tabelas e campos, além de efetuar a criação e manutenção dos OBJETOS nos bancos de dados (TRIGGERs e PROCEDUREs);
    - **[PASSWORD]**: Senha de acesso ao banco de dados;
    - **[OWNER]**: Nome do usuário proprietário das tabelas que farão parte do projeto de integração. Em alguns bancos e ambientes não é necessário informar este campo, pelo fato do usuário informado em **[USER NAME]** ser por padrão o proprietário dos OBJETOs;
    - **[Inserir database como prefixo nos OBJETOS]**: Selecione esta opção para que, dentro da TRIGGER quando for efetuada uma referencia a algum OBJETO, seja inserido o nome do database como PREFIXO destes OBJETOS. Esta opção é selecionada quando é necessário acesso a servidores distintos na integração entre dados de ORIGEM e DESTINO.

Anotações:

- **[Apresentar VIEWs junto com as tabelas]**: (Somente para conexão com ORACLE) Em alguns casos, faz-se necessário a criação de TRIGGERs (integração) com ORIGEM em uma VIEW ou alguma referencia de DESTINO com VIEWs, para isto, foi disponibilizada a opção de apresentar na lista de OBJETOS as VIEWs.

Após a configuração de todos os campos desta tela, deve-se pressionar o botão **[CONECTAR]** (em Azul). Neste momento a ferramenta fará algumas ações como descrito abaixo:

- Efetuará a conexão com o banco de dados ORIGEM;
- Efetuará a conexão com o banco de dados DESTINO;
- Fará a validação da LICENÇA de uso da ferramenta;
- Buscará e disponibilizará todas as tabelas dos bancos de dados de ORIGEM e DESTINO;
- Gravará os dados de conexão no arquivo .INI;
- O botão **[CONECTAR]** será substituído pelo botão **[DESCONECTAR]** (em vermelho).

Estes passos são executados nesta sequencia sendo que se em alguma das etapas ocorrer algum problema as demais etapas não serão executadas.

Alguns dos erros mais comuns são a não informação ou informação errada referente ao OWNER da conexão. Quando ocorrer algum problema durante o processo de conexão, efetue a conexão ao banco dados com outra ferramenta e certifique-se de que todas as informações foram digitadas corretamente.

Para outras situações entre em contato com nosso suporte pelo endereço: suporte@ii2.com.br.

Criando integrações

Após conectar-se aos bancos de dados ORIGEM e DESTINO pode-se iniciar os trabalhos de criação das integrações. O ideal é que o profissional que for efetuar esta atividade tenha conhecimento da estrutura de ambos os bancos de dados, facilitando assim o desempenho desta atividade.

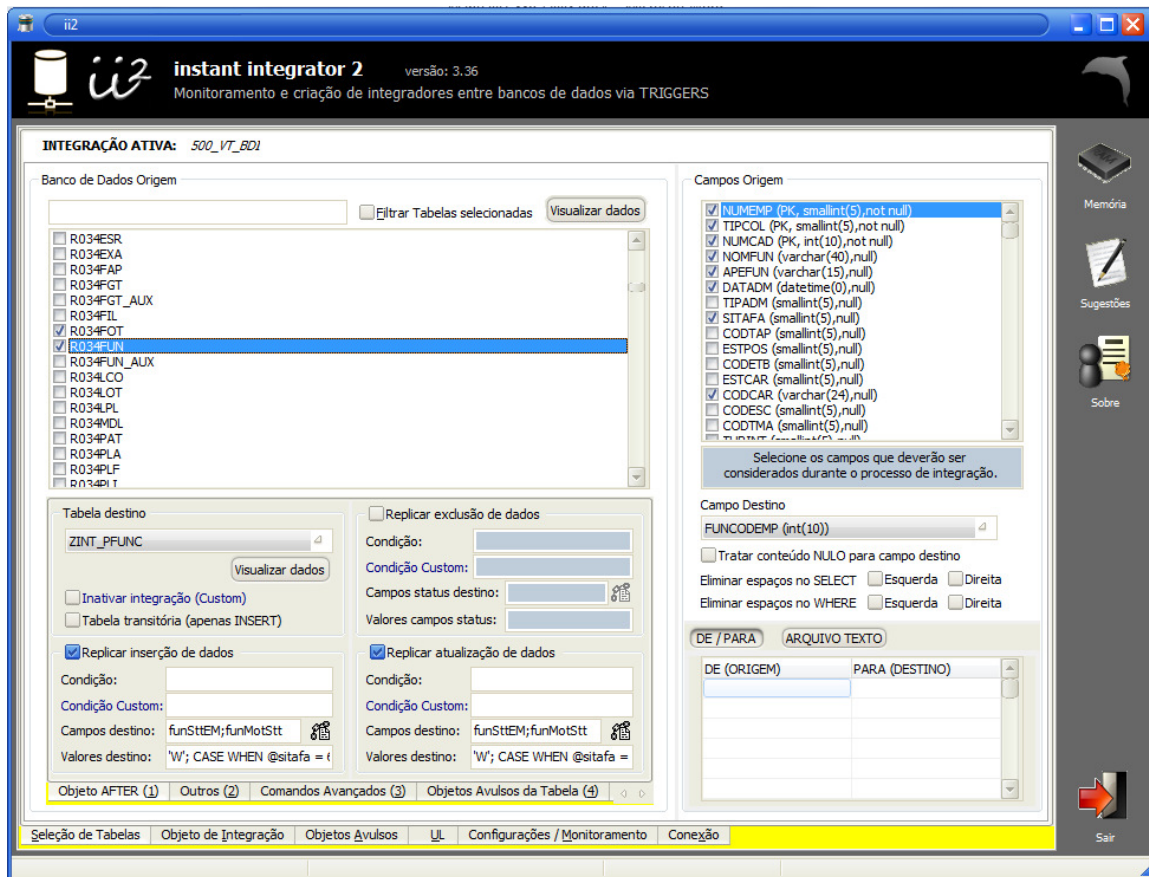
Sendo assim, na ABA Seleção de tabelas, deverá aparecer uma lista todas as tabelas existentes no banco de dados de ORIGEM. Selecione a tabela que irá

Anotações:

fazer parte do seu PROJETO de integração e marque o checkbox a sua frente. Em seguida, selecione a tabela de DESTINO no combo logo abaixo da lista. Desta forma já temos a referencia entre a tabela de ORIGEM e DESTINO, selecione então quais tipos de integrações serão consideradas nesta interface, selecionando o checkbox nas opções:

- Replicar inserção de dados;
- Replicar exclusão de dados;
- Replicar atualização de dados.

Com isto, só resta fazer o kink entre os campos das tabelas de ORIGEM e DESTINO e sua primeira integração estará finalizada. Para fazer o link entre campos basta selecionar os campos da tabela de ORIGEM no lado direito da tela, marca-los como fazendo parte da integração e em seguida, para cada campo marcado, selecionar qual o seu campo correspondente da tabela de DESTINO no combo logo abaixo a lista.

Anotações:

Durante o processo de ligação entre os campos de ORIGEM e DESTINO, a ferramenta automaticamente faz alguns ajustes para evitar problemas na criação da TRIGGER. Alguns destes ajustes automáticos são:

- Adequação do tamanho dos dados campo de ORIGEM para o campo de DESTINO quando este for menor, evitando assim o estouro no tamanho dos dados;
- Conversão do tipo de dados quando for efetuado um link entre campos de tipos diferentes (VARCHAR para INT, DATETIME para VARCHAR etc)

Alguns outros tratamentos que podem ser selecionados no tratamento dos campos são os checks que estão logo abaixo do combo dos campos de DESTINO:

Anotações:

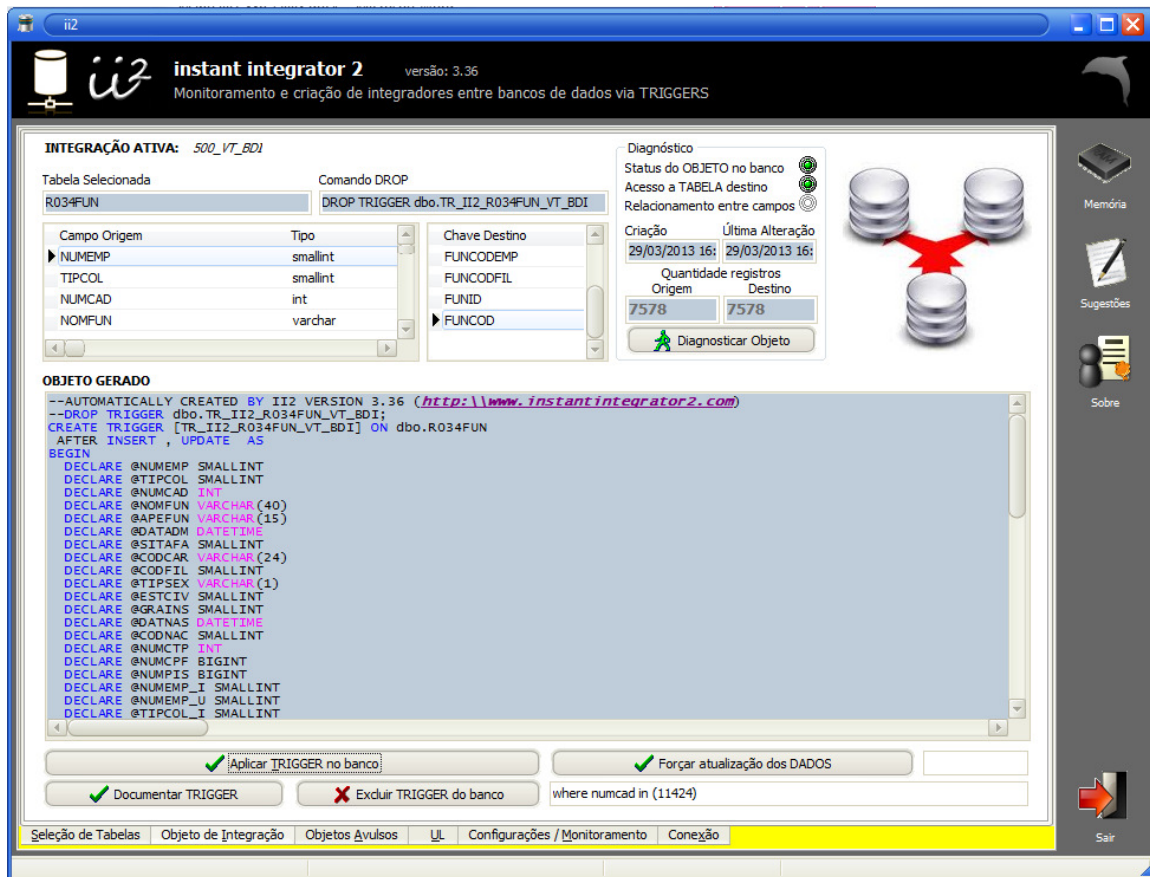
- Tratar conteúdo NULO para campo destino: Esta opção quando selecionada impede que seja gravado o valor NULL no registro de DESTINO, enviando assim, um espaço quando o campo for do tipo VARCHAR, zero quando o campo for do tipo NÚMERO e a primeira data válida (depende de cada tipo de banco de dados) para quando o campo for DATA;
- Eliminar espaços brancos a esquerda do destino, Eliminar espaços brancos a direita do destino: Esta opção elimina os espaços que houver nos registros de ORIGEM quando forem enviados para o DESTINO.
  - Em alguns tipos de bancos de dados é disponibilizada a opção de se tirar os espaços da esquerda e/ou da direita do conteúdo do campo, em outros bancos de dados, só é habilitada a opção de se tirar os espaços da direita **"e"** esquerda ao mesmo tempo.

Finalizada esta parametrização, selecione a ABA Objeto de Integração e já será possível visualizar a TRIGGER montada podendo assim ser aplicada no banco de dados.

- Lembre-se que todos os OBJETOS de banco de dados criados e gerenciados pela ferramenta II2 estão sempre relacionados ao banco de dados ORIGEM de cada PROJETO. Caso seja necessário a intervenção em algum OBJETO que esteja alocado no banco de dados de DESTINO será necessário a criação de um segundo PROJETO de integração invertendo a posição dos bancos de dados (ORIGEM para DESTINO e DESTINO para ORIGEM).

Na ABA Objeto de Integração ainda é apresentado os campos que fazem parte da chave primária (PRIMARY KEY) tanto da tabela de ORIGEM como de DESTINO para facilitar a consulta destes durante a montagem de integrações mais complexas onde haja a necessidade de troca de chave (como veremos em tópicos mais a frente). Podemos também fazer algumas interações com a TRIGGER gerada pela ferramenta como veremos a seguir:

- Diagnosticar OBJETO: Verifica se a TRIGGER já está criada no banco de dados, apresentando sua data de criação e a data da ultima atualização ocorrida nela. Apresenta ainda a quantidade de registros existentes na tabela a qual ela é referenciada tanto na ORIGEM como no DESTINO para que se possa analisar de há uma diferença na quantidade de registros indicando desta forma algum problema de sincronismo de dados nesta interface;

Anotações:

- Aplicar TRIGGER no banco: Faz a criação da TRIGGER no banco. Caso ocorra algum problema neste processo, será apresentada uma mensagem com o erro que ocorreu para que possa ser tratado pelo usuário;
- Documentar TRIGGER: Gera um arquivo .TXT com o código gerado para a TRIGGER. O caminho (pasta) onde será gravado o arquivo pode ser parametrizado na ABA Monitoramento no campo "Pasta para documentação";
- Excluir TRIGGER do banco: Apaga a TRIGGER do banco de dados;
- Forçar atualização dos dados: Força a execução de um UPDATE em todos os registros da tabela de ORIGEM, executando desta forma a TRIGGER de UPDATE que fará com que os dados sejam enviados a tabela de DESTINO automaticamente.

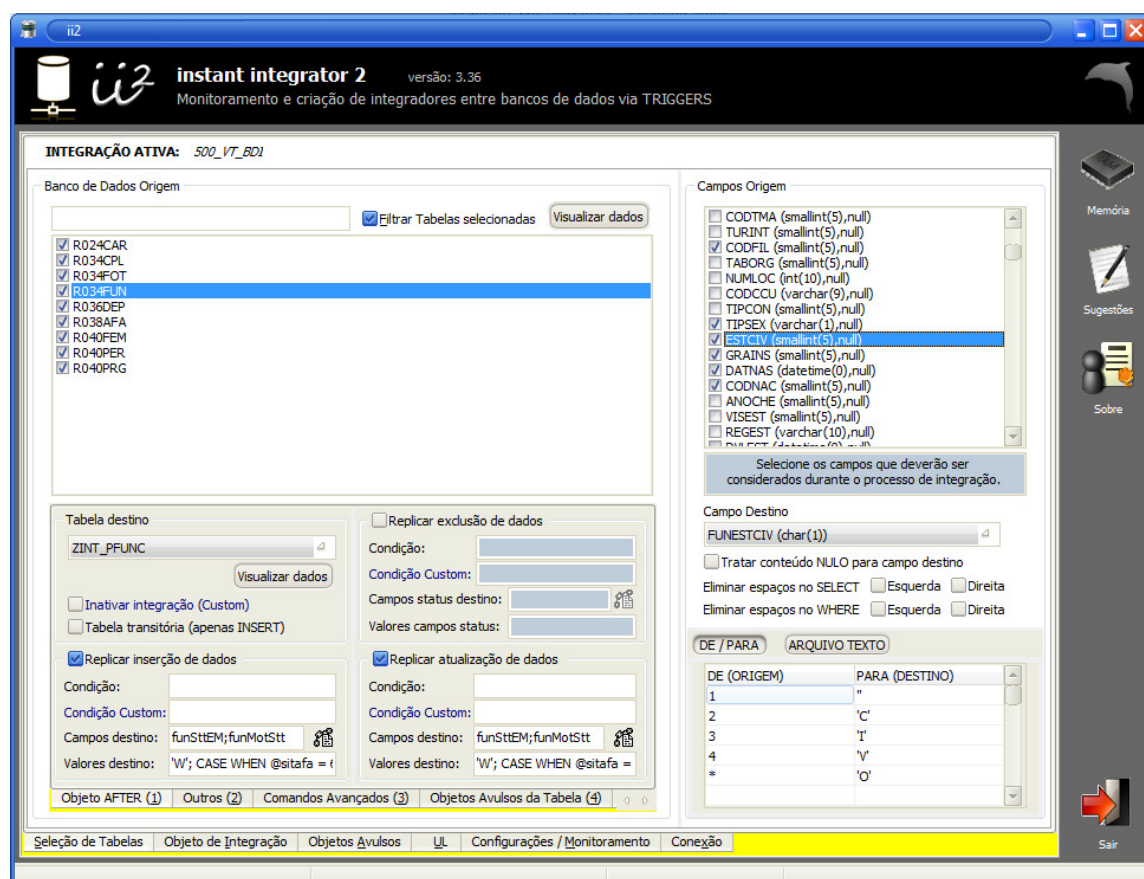
Anotações:

- A opção de forçar atualização de dados somente estará habilitada caso seja seleciona a opção Replicar Atualização de Dados na ABA Seleção de Tabelas;
- Para forçar a atualização de dados é possível incluir uma cláusula WHERE no campo logo abaixo do botão, facilitando assim a filtragem de dados e a execução desta rotina apenas para uma seleção específica de dados;
- Durante a execução desta rotina será apresentada uma barra de progressão para acompanhamento da evolução dos registros processados com tempo estimado para término. Caso ocorra algum erro durante a progressão e execução do comando em algum registro, será apresentada uma mensagem informando o erro retornado pelo banco de dados podendo o comando executando ser copiado para a memória para posterior análise em uma ferramenta específica do banco de dados. Pode-se ainda cancelar a continuação deste processo, ou solicitar que não seja mais apresentada a mensagem de erro quando este ocorrer, todavia, ao término do processo será apresentado um resumo com a quantidade de registros que foram processados e quantos destes registros apresentaram erros durante o processo;
- Esta rotina somente terá efeito caso a TRIGGER já tenha sido aplicada ao banco de dados, de outra forma, esta rotina será executada, porém não surtirá efeito algum no banco de dados, uma vez que o comando UPDATE executado faz a atualização do valor de um determinado campo (que não seja uma chave primária) com o seu próprio conteúdo.

Durante o processo de implantação de um projeto de integração, é possível ir ligando a integração das tabelas gradualmente, para isto, foi criado a opção Inativar Integração (Custom), facilitando assim a Aplicar personalização de TRIGGERS na Aba Monitoramento.

Anotações:

Trabalhando com lista DE/PARA

É possível alterar as informações de ORIGEM ao serem enviadas ao DESTINO utilizando se da lista de DE/PARA constante na ABA de **[Seleção de Tabelas]** conforme demonstrado abaixo:

Para utilização da lista DE/PARA selecione primeiro a tabela ORIGEM linkando-a a uma tabela de DESTINO. Em seguida selecione o campo de ORIGEM linkando- a um campo de DESTINO e em seguida preencha a lista DE/PARA com os valores que serão transformados em sua ORIGEM para um novo valor em seu DESTINO, conforme na imagem acima segue:

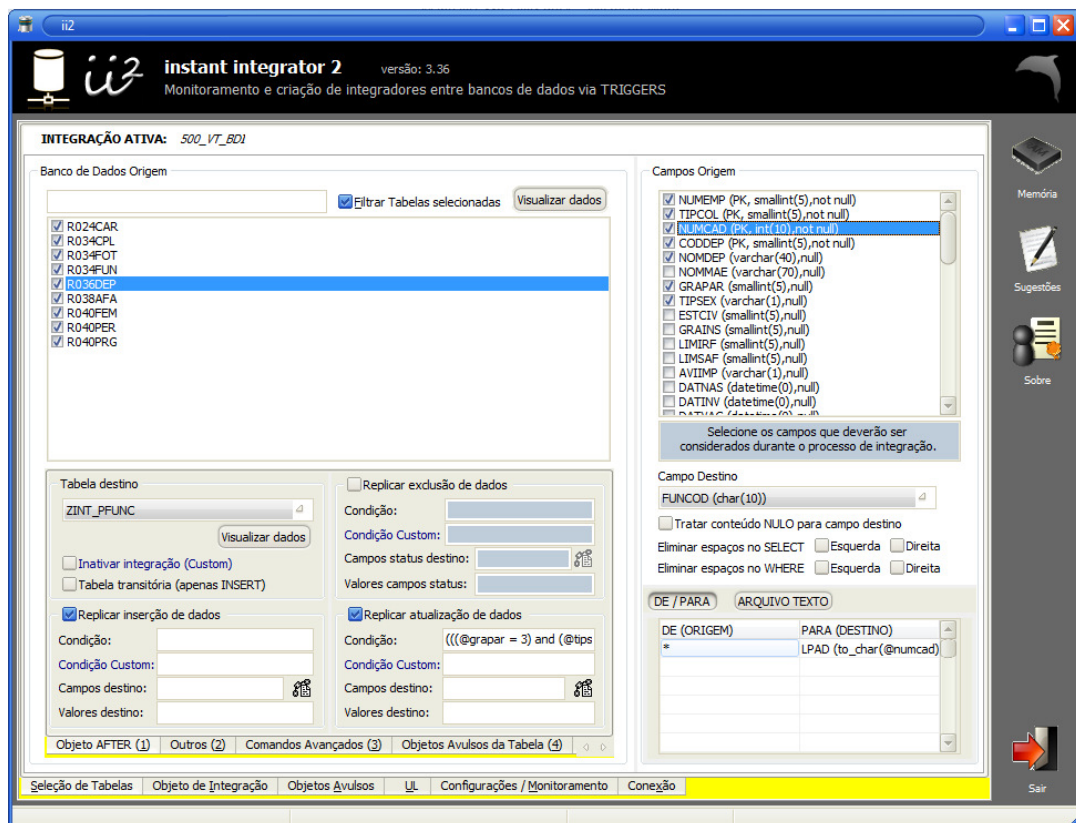
- Se o valor na ORIGEM for igual a 1 então será enviado ao DESTINO ‘’;
- Se o valor na ORIGEM for igual a 2 então será enviado ao DESTINO ‘C’;

Anotações:

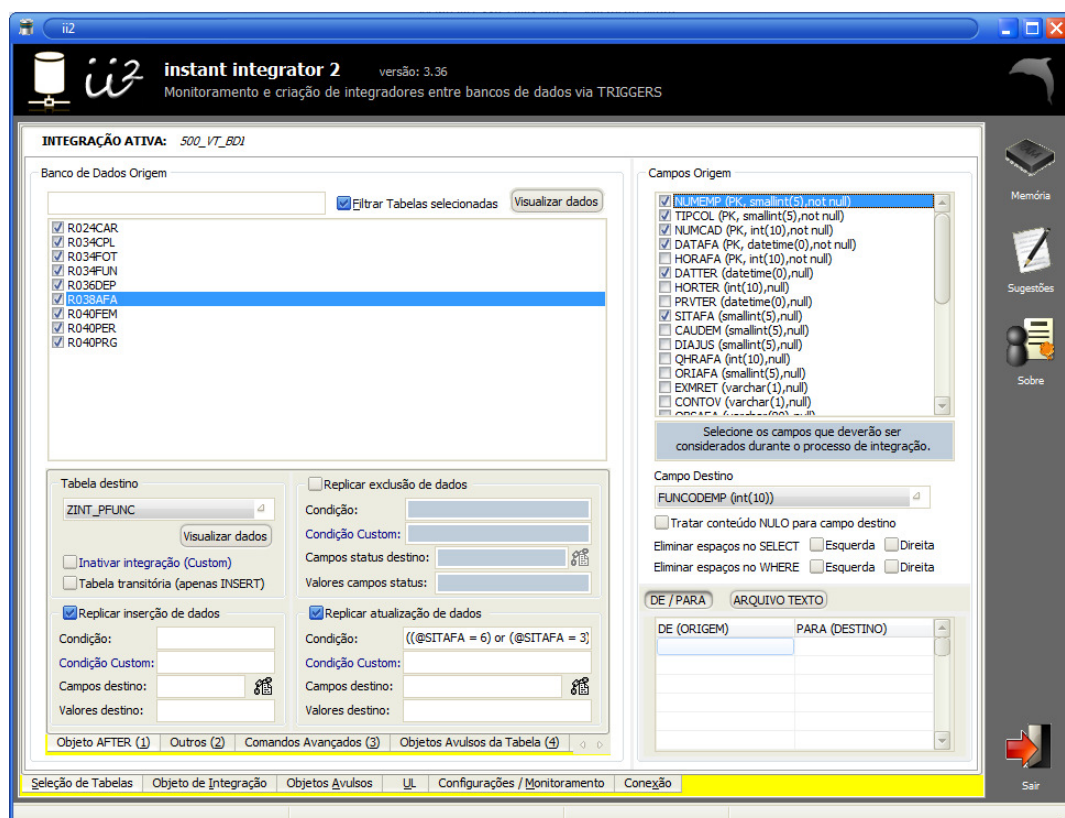
- Se o valor na ORIGEM for igual a 3 então será enviado ao DESTINO 'I';
- Se o valor na ORIGEM for igual a 4 então será enviado ao DESTINO 'V';

Note que o link entre os campos de ORIGEM e DESTINO foi efetuado com tipos de campos diferentes, ou seja, de um campo do tipo SMALLINT para um campo do tipo CHAR.

- Também foi inserida uma exceção de valores, onde na última linha da lista de DE/PARA utilizou-se os seguintes valores: * e 'O' que terá a seguinte funcionalidade:
  - Se o valor na ORIGEM for qualquer um diferente dos casos já citados (1, 2, 3 ou 4) então envie para o destino o valor 'O'.

É possível ainda montarmos valores para o DESTINO ou mascara de campos para atender as mais diversas necessidades de integração de dados entre bancos distintos utilizando-se das listas DE/PARA. Observe o exemplo a seguir:

 Anotações:

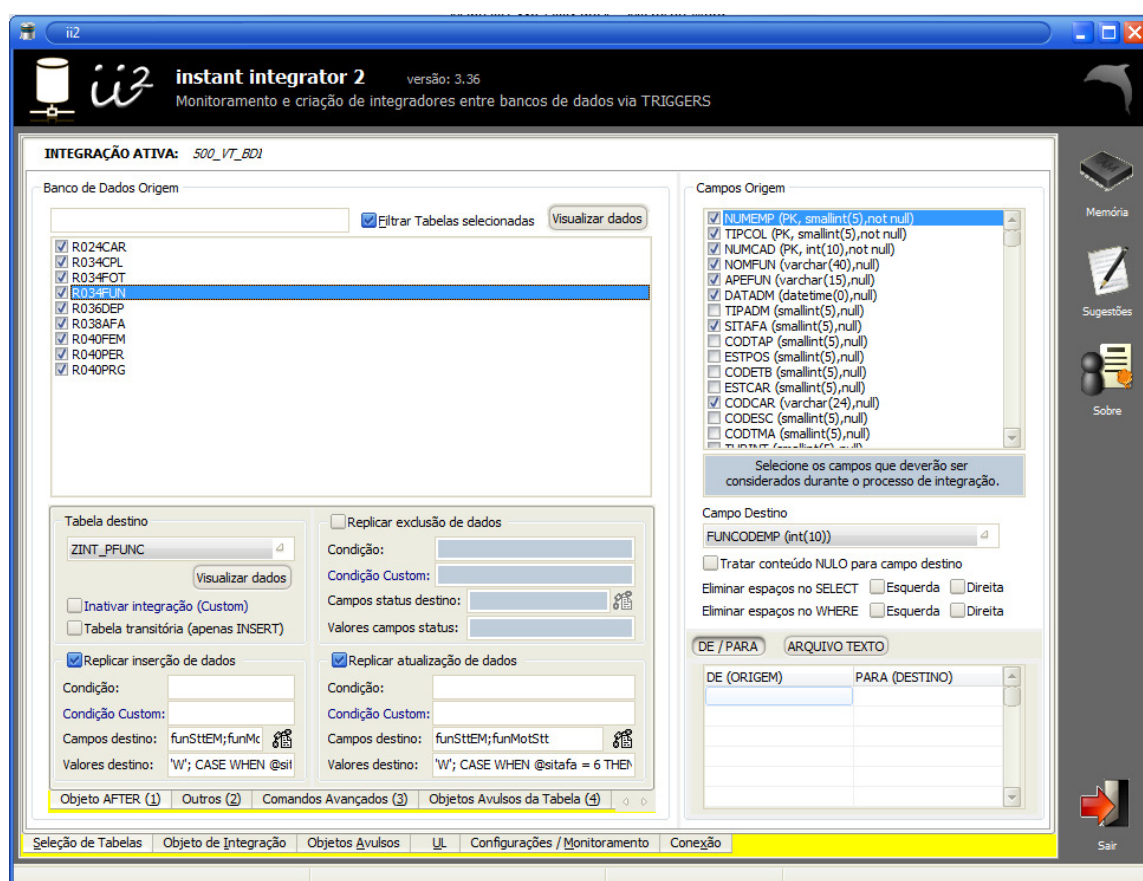
Condições de execução

Para cada uma das situações de tipos de TRIGGERS que forem selecionadas (INSERT, UPDATE ou DELETE) é possível se condicionar a alguma situações ou cláusula distinta, criando assim uma integração de dados apenas para uma determinada seleção conforme explanaremos com base na tela abaixo:

Note que na opção de atualização de dados (UPDATE) foi inserida uma condição no campo logo abaixo ao check do mesmo. Só serão atualizados os registros cujo campo SITAFA for igual a 6 ou igual a 3.

Lembre-se que ainda há a opção de se criar condições customizadas para cada ambiente distinto (estas condições são salvas em arquivos INI separados), conforme exemplo acima, neste ambiente (conjunto de servidores) só serão atualizados os registros cujo campo NUMEMP dor igual a 6.

 Anotações:

Campos DESTINO sem ORIGEM

Existem casos em que campos na TABELA destino ocorra a necessidade de se preencher com um determinado valor (não podendo o mesmo ser preenchido com um valor NULO para que não perca a integridade das informações ou pelo motivo do campo ser do tipo NOT NULL) ou, ainda que possua um campo de ORIGEM seja necessária a troca do valor ou ainda o envio de um valor fixo.

Para isso, foram criados os campos: “Campos destino” e “Valores destino” nas cláusulas INSERT e UPDATE. Neles é possível informar os campos da tabela de DESTINO e os valores que se deseja inserir nos mesmos (podendo desta forma efetuar combinação de valores e utilização de funções do banco de dados e comandos avançados). Para se informar mais de um campo, deve-se separá-

Anotações:

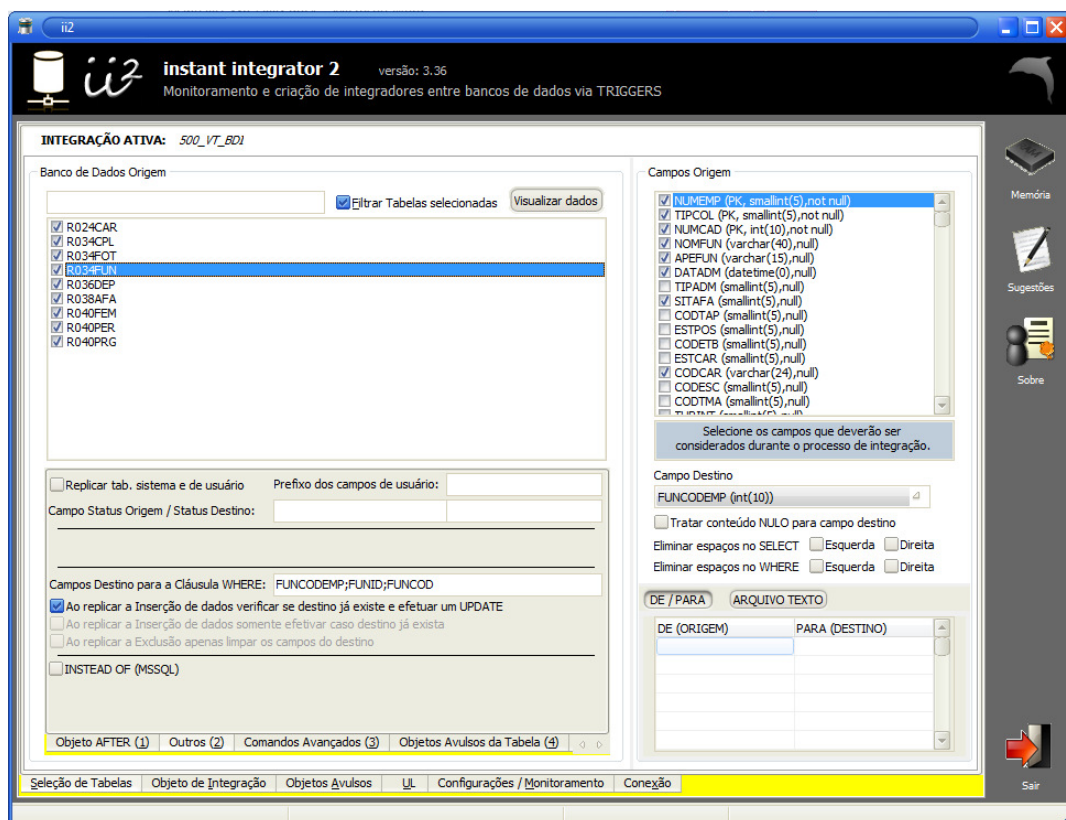
los com um ponto e vírgula (;). A quantidade de campos informado em Campos destino deve ser a mesma quantidade de campos informado em Valores destino, de outra forma a ferramenta apresentará uma mensagem de erro no momento da criação da TRIGGER.

Note na tela acima, na clausula UPDATE foram inseridos dois campos (separados por ponto e vírgula) e no campo de valores foi inserido um valor fixo e uma condição CASE que é a situação que poderia ser tratada pela tabela DE/PARA caso houvesse um campo de ORIGEM relacionado a ele.

Anotações:

Alterando a chave primária DESTINO (PK)

Em algumas situações específicas, é necessário alterar os campos de consulta para validação de chave primária (PRIMARY KEY) na tabela de DESTINO. Durante um processo e INSERÇÃO, ATUALIZAÇÃO ou EXCLUSÃO de dados, estes campos são essenciais para consultar se o registro já existe ou não na tabela de DESTINO, todavia, há casos em que a chave primária não representa o elo entre a tabela de ORIGEM e DESTINO como, por exemplo: Num cadastro de cliente a chave primária no DESTINO é o ID do cliente proprietário do sistema, porém este ID não existe na ORIGEM sendo necessária a troca do elo entre as tabelas, fazendo-se a consulta e link pelo campo CNPJ.

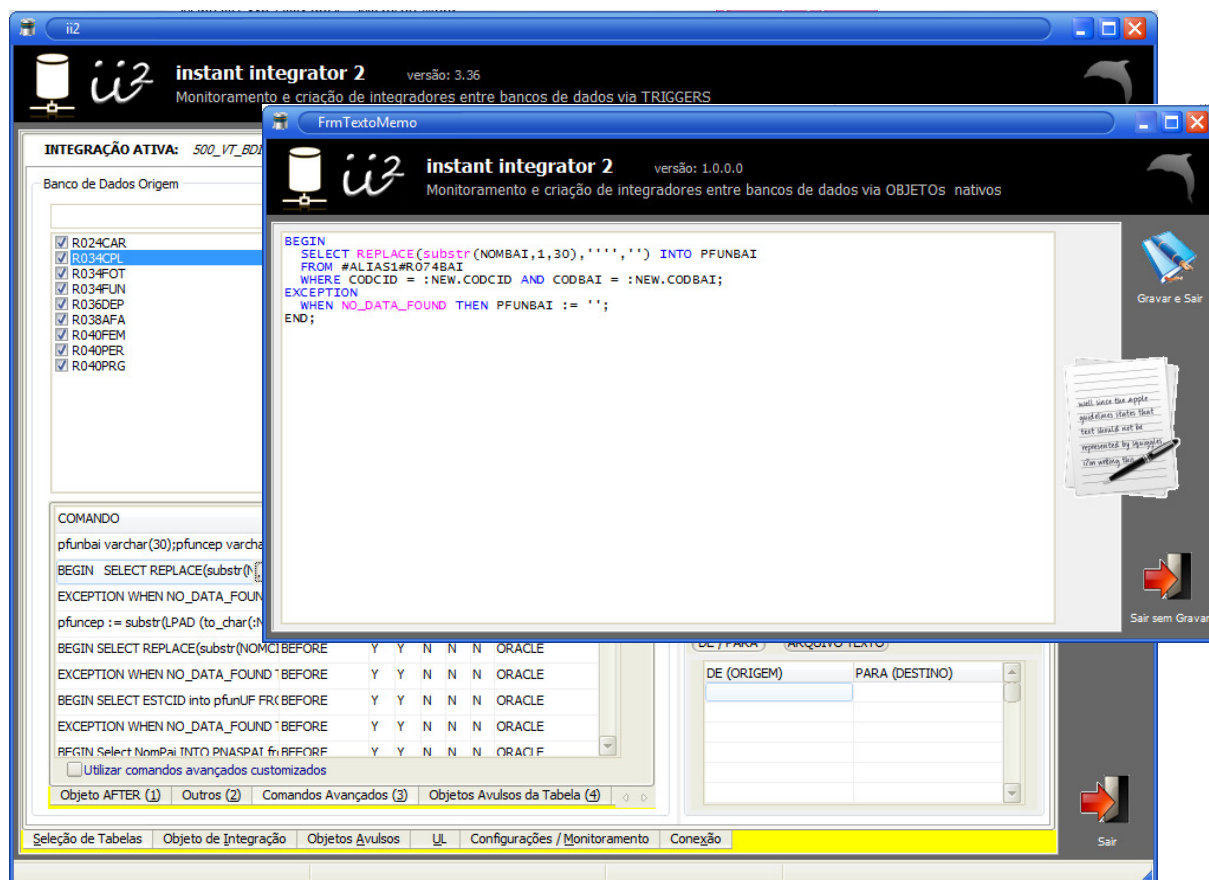
Para atender esta necessidade foi disponibilizada na ABA outros dentro da tela de Seleção de Tabelas o campo: "Campos Destino para a Cláusula WHERE". Para informar mais de um campo como chave (chave composta) os mesmos devem ser separados por um ponto e vírgula (;). Veja o exemplo a seguir:

Anotações:

Comandos Avançados

Para atender a necessidade de integrações mais complexas, foi disponibilizada a opção de inserir comandos mais avançados de código PLSQL em determinados pontos pré-definidos da TRIGGER que será gerada.

Na ABA Comandos Avançados na tela de Seleção de Tabelas, é possível a inserção dos comandos e setar onde as mesmas serão inseridas no código conforme abaixo:

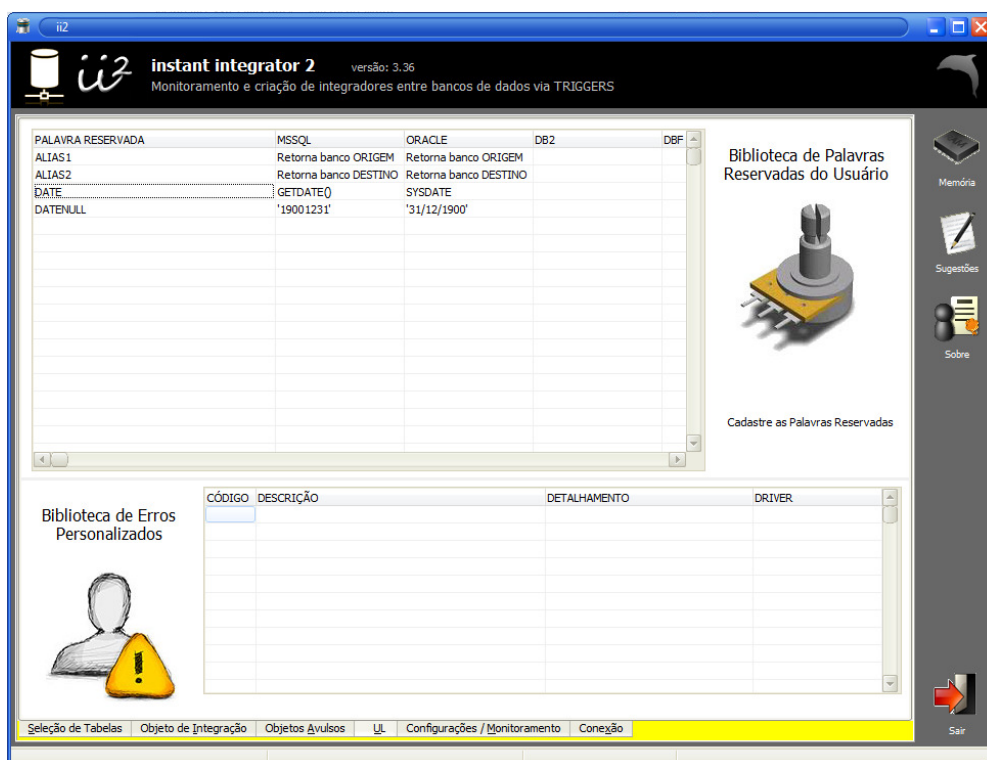
- ANTES ou DEPOIS do comando INSERT;
- ANTES ou DEPOIS do comando UPDATE;
- ANTES ou DEPOIS do comando DELETE;
- DEPOIS da cláusula DECLARE;
- ANTES ou DEPOIS do comando SELECT onde é igualada as variáveis para serem utilizadas na TRIGGER.

Anotações:

Para atender a portabilidade dos comandos em bancos de dados distintos, foi criada a opção de selecionar em qual tipo de banco será aplicado o comando avançado, tornando assim mais abrangente a utilização dos comandos avançados.

Nos comandos avançados ainda é possível a utilização de comandos da biblioteca de usuários também com a opção de portabilidade de tipos de bancos de dados como veremos no próximo tópico.

Anotações:

Trabalhando com biblioteca de USUÁRIOS

É possível a criação de palavras chaves para serem trocadas de acordo com o tipo de banco de dados selecionado de acordo com o ambiente em que se estiver trabalhando. Desta forma garantimos a portabilidade de bancos de dados na troca de ambientes.

A biblioteca de usuários (USER LIBRARY) pode ser utilizada em diversos locais da ferramenta, sendo:

- Condições de execução;
- Comandos avançados;
- Tabela DE/PARA.

Anotações:

Parâmetros de inicialização do II2

Alguns parâmetros foram criados para facilitar o manuseio dos arquivos de projeto na utilização do II2. Com eles será possível a flexibilização do uso da ferramenta como será descrito abaixo:

- **-i** utilizado para efetuar a leitura dos arquivos .INI numa pasta diferente da pasta principal onde o .EXE se encontra. Com este parâmetro é possível a divisão de grupos de projetos em subpastas facilitando assim a organização dos arquivos para posterior distribuição dos mesmos.
  Sua utilização deve se dar da seguinte forma:
  II2.EXE –i\PROJETO1\
  Desta forma o II2 será executado na pasta onde estiver copiado, por exemplo: "C:\INTEGRADOR\" e os arquivos .INI que contém as informações dos projetos serão lidos da pasta: "C:\INTEGRADOR\PROJETO1\"

- **-p** utilizado para entrar no II2 já com um projeto carregado e conectado. Utilizado para entrar na ferramenta automaticamente ativando-se o monitoramento do banco de dados e manter a integridade do projeto e dos dados entre o banco de dados de ORIGEM e DESTINO.
  Sua utilização deve se dar da seguinte forma:
  II2.EXE –pII2_PROJETO1.INI
  Desta forma o II2 será executado já conectado ao projeto [PROJETO1] e estando com a opção de monitoramento ativada e a opção de execução em modo MINIMIZADO ativado, entra já monitorando toda integração parametrizada.

  - É possível também utilizar a combinação destes parâmetros na execução do II2, sendo assim, utilizado da seguinte forma:
    II2.EXE –i\PROJETO1\ –pII2_PROJETO1.INI
    Desta forma, o II2 executará conectado ao [PROJETO1] dentro da pasta "C:\INTEGRADOR\PROJETO1\" considerando o exemplo acima onde o II2.EXE encontra-se na pasta "C:\INTEGRADOR\".

Anotações:

Releases da Ferramenta

VERSÃO: 3.03 (27/6/2011)

- ✓ Criação de SIGLA para a integração que será utilizado no nome das TRIGGERs para possibilitar a criação de mais de uma TRIGGER na mesma tabela por integrações diferentes
- ✓ A sigla passa a ser utilizada no nome dos CURSORES para que não ocorra erros de 2 Objetos sendo executados ao mesmo tempo com o mesmo nome
- ✓ Opção de inserir o DATABASE NAME como prefixo dos OBJETOS de origem e destino possibilitando o acesso a objetos por um USER diferente aos OWNER dos objetos
- ✓ Criado campos para inserção de comentários tanto para a integração como para cada OBJETO individualmente
- ✓ Alterado o layout da Tela1 colocando um PageControl para separar o objeto AFTER, BEFORE e Comentários
- ✓ Colocado dimensionadores nas telas para possibilitar aos usuários o ajuste dos componentes
- ✓ Montagem de CASE na versão SQL para a lista de DE/PARA

VERSÃO: 3.04 (29/6/2011)

- ✓ Quando não selecionada a chave de destino, está declarando variáveis com apenas @_I. O sistema passa a procurar os valores da chave nos campos de INSERT e VALORES
- ✓ Alterado o layour da Tela1 para tornar mais Friendly a utilização de Insert, Update e Delete
- ✓ Inserido campos para tratar valores de campos destinos no UPDATE assim como eram tratados para os INSERTs
- ✓ Trocado de lugar o tratamento do DE/PARA. Ao invés de ser tratado no SELECT dos campos ORIGEM, está sendo tratado ao gravar os campos DESTINO
- ✓ Inserido campo para informar os campos que poderão ser utilizados como chave para montagem das CLAUSULAS WHERE dentro dos objetos gerados
- ✓ Quando for utilizada uma máscara ou o recurso de DE/PARA não faz o CAST
- ✓ Fixado o nome da variavel que irá receber o retorno do cursor com o conteudo do primeiro campo chave da tabela destino, pois quando este campo não era selecionado, o cursor apresentava um erro

VERSÃO: 3.05 (1/7/2011)

- ✓ Acertado o campo de condições (INSERT, UPDATE e DELETE) que quando estava sendo utilizado estava substituindo a condição de entrada nos tipos de replicação
- ✓ Criado uma Checkbox para permitir que o INSERT faça um UPDATE caso encontre a chave
- ✓ Trocado o caracter divisor de campos de ',' para ';' nas entradas "Campos e Valores dos Inserts", "Campos e Valores dos Updates", "Campos Where" e "Campos de Status"
- ✓ Inserido no sistema algumas palavras chaves para serem substituidas no momento da criação dos Objetos
- ✓ As palavras chaves serão utilizadas nos campos de condição (INS, UPD, DEL), nos campos de valores (INS, UPD)
- ✓ Palavra chave criada: #DATE# - Substitui pela função que pega a data atual do sistema de acordo com o banco conectado
- ✓ Palavra chave criada: #ALIAS1# - Substitui pelo Database e User configurado como Origem
- ✓ Palavra chave criada: #ALIAS2# - Substitui pelo Database e User configurado como Destino

VERSÃO: 3.06 (5/7/2011)

- ✓ Igualado os comandos de SELECT para pegar os valores antigos dos campos (INSERT, UPDATE, DELETE) pois o SELECT da Clausula DELETE estava igualando apenas os campos CHAVES
- ✓ Acertado BUG que quando utilizava a opção de "AO INSERIR EFETUAR O UPDATE" estava criando uma condição que só entraria em caso de UPDATE e o INSERT não estava funcionando

Anotações:

- ✓ Quando documenta um Objeto, o arquivo gerado passa a ter o nome da Integração + Nome da Tabela
- ✓ A tabela DE/PARA passa a considerar as palavras chaves para efeito de troca
- ✓ Criada opção de forçar carga de dados, onde será dado um UPDATE registro a registro na tabela ORIGEM forçando a execução do OBJETO
- ✓ Alterado o local onde é colocado a condição do INSERT para ficar após o SELECT que iguala os valores das variáveis aos valores dos campos
- ✓ Acertado campos tipo MEMO não estavam recuperando mais de uma linha quando gravadas desta forma
- ✓ Acertado campos que não estavam recuperando as apóstrofes quando lidas do arquivo INI

VERSÃO: 3.07 (13/7/2011)

- ✓ Eliminado o tratamento via CAST de campo CHAR para VARCHAR e vise-versa
- ✓ Criado campo para complementar comando utilizado para forçar carga de dados. Informações informadas não serão validadas, apenas serão inseridas no SELECT da tabela
- ✓ Inserido a opção de registrar uma PROCEDURE por campo destino a ser utilizado
- ✓ Inserido no inicio dos OBJETOS a chamada as procedures registradas
- ✓ Eliminado a gravação de arquivo XML para gravação dos dados de conexão com os bancos de dados, sendo estes tratados apenas pelos arquivos INI
- ✓ Gravação de dados de conexão passa a ocorrer após a conexão com o BANCO
- ✓ DECLARE e comparação de informações para UPDATE não estava considerando o campo WHERE para montagem da chave, o que estava causando problema quando a chave destino não estava sendo utilizada na LISTA de campos origem

VERSÃO: 3.08 (20/7/2011)

- ✓ Criado tela para acompanhamento da evolução da aplicação com todas as funcionalidades que são implantadas
- ✓ Ajustada a leitura do projeto já criado que solicitava duas vezes para efetuar a leitura do arquivo INI
- ✓ Ajustado campos cadastrados na cláusula WHERE para buscar Origem do campo com valores customizados na opção do UPDATE e se ainda não encontrar busca na opção do INSERT
- ✓ Palavra chave criada: #DATENULL# - Substitui pela literaria da primeira data valida do banco de acordo com a conexão parametrizada
- ✓ Ajustada a condição para execusão do comando DELETE que estava antes do SELECT dos campos da tabela
- ✓ Criado opção de gerar comandos avançados para serem inseridos dentro do Objeto Gerado de acordo com a necessidade e local necessário. A única validação que será executada é a troca de palavras reservadas
- ✓ Criada opção de filtrar apenas as tabelas que estão marcadas para a integração, para melhorar a visualização das integrações

VERSÃO: 3.09 (27/7/2011)

- ✓ Criada coluna DECLARE nos comandos avançados para inserção de comandos na sessão de declaração de variáveis, as quais poderão ser utilizadas com os próprios comandos avançados inseridos em outras sessões
- ✓ Retirado o campo de TRIGGER customizada para evitar problemas de funcionamento dos OBJETOS na aplicação
- ✓ Criada coluna SELECT nos comandos avançados para inserção de comandos antes e depois do SELECT para buscar variáveis que estão sendo utilizadas do registro alterado
- ✓ Criada funcionalidade de se efetuar um TRUNC do valor do campo quando o tamanho do campo ORIGEM for maior que o tamanho do campo DESTINO. (Apenas para campos CHAR e VARCHAR)
- ✓ Alterado o momento da criação da TRIGGER que deixou de ser criada ao se clicar na tabela de ORIGEM e passou a ser criada no momento que necessário para a analise e criação da mesma no banco de dados

✍ Anotações:

- ✓ Ajustado BUG que estava setando campo de ORIGEM quando possuia o mesmo prefixo de outros campos com o mesmo nome
- ✓ Alterado posição do CLOSE CURSOR que poderia gerar algum problema quando se trabalhava com TRIGGERs recursivas
- ✓ Alterado posição dos botões da página principal

VERSÃO: 3.10 (1/8/2011)

- ✓ Ajustado e homologado a geração de objetos no ORACLE
- ✓ Inserido coluna de Driver na GRID de OBJETOS AVULSOS para suportar os diversos tipos de banco de dados
- ✓ Inserido coluna de Driver na GRID de COMANDOS AVANÇADOS para suportar os diversos tipos de banco de dados
- ✓ Ajustado problema na declaração das variáveis que não estava explicitando o tamanho dos campos e gerando problema de arredondamento dos valores
- ✓ Ajustado comparação de tipos de campos para que não ocorra CAST entre campos do tipo MONEY, SMALLMONEY e DECIMAL
- ✓ Ajustado mecanismo de conexão com banco de dados diferentes que em algumas situações causava erro de conexão com os bancos

VERSÃO: 3.11 (8/8/2011)

- ✓ Criado conceito de palavras reservadas dos usuários, sendo salvo em um arquivo nomeado ii2_UserLibrary.INI que servirá para todos os projetos. As palavras reservadas deverão ser relacionados seus valor para cada banco de dados disponíveis para conexão
- ✓ Objetos Avulsos começam a passar por função de troca de palavras chaves
- ✓ Ajustado busca de campos chave no banco de dados ORACLE

VERSÃO: 3.12 (15/8/2011)

- ✓ Alterado layout das telas para melhor entendimento e facilidade na utilização das rotinas básicas do sistema
- ✓ Ajustado comparação de tipos de campos para que não ocorra CAST entre campos VARCHAR2 e CHAR (ORACLE)
- ✓ Ajustado identação de geração de OBJETOS em Oracle

VERSÃO: 3.13 (26/8/2011)

- ✓ Criado botão para DROP da TRIGGER que está selecionada
- ✓ Ajustes na geração dos OBJETOS em Oracle
- ✓ Ajustado Forçar Carga para OBJETOS em Oracle
- ✓ Ajustado o tamanho dos campos das tabelas em Oracle
- ✓ Alterada a mensagem apresentada quando ocorre algum erro no banco durante o processo de FORÇAR CARGA para monstar a mensagem de erro original retornada pelo banco de dados
- ✓ Substituído a utilização dos comandos DBMS_SQL por EXECUTE IMMEDIATE na geração de OBJETOS em ORACLE

VERSÃO: 3.14 (7/9/2011)

- ✓ Ajustado a CLAUSULA Where para campos do tipo DATE na geração de OBJETOS em Oracle
- ✓ Ajustado o tratamento para campos com conteúdo NULO na geração de OBJETOS em Oracle
- ✓ Opção de cancelar a atualização de dados quando é exibida a mensagem de ERRO em algum registro
- ✓ Ajustado replicação de campos tipo valor com casas decimais na geração de OBJETOS em Oracle
- ✓ Ajustado a rotina de Forçar Atualização de Dados para tabelas que possuam campos do tipo DATE ou DATETIME em sua chave primária

Anotações:

- ✓ Criada opção para trabalhar com VIEWs com características de TABLEs para geração de OBJETOS em ORACLE

VERSÃO: 3.15 (17/9/2011)

- ✓ Criado botão para enviar para o clipboard o comando executado quando ocorrer uma mensagem de erro durante procedimento de Forçar Atualização
- ✓ Criado menu POPUP no script de geração da TRIGGER com opção de copiar o código para o clipboard
- ✓ Criado menu POPUP em OBJETOS avulsos para copiar o código para o clipboard
- ✓ Adicionado validação e mensagem para usuário quando selecionada uma tabela de ORIGEM e não for identificada a tabela parametrizada como DESTINO
- ✓ Adicionado validação e mensagem para usuário quando selecionado um campo de ORIGEM e não for identificado o campo parametrizado como DESTINO

VERSÃO: 3.16 (27/9/2011)

- ✓ Criado painel de diagnóstico na tela de OBJETO gerado para verificação de consistências para a interface gerada
- ✓ Criado parâmetro de entrada do executável "-i" para leitura de arquivos INI em subpasta da pasta principal de onde se está executando o aplicativo
- ✓ Criado parâmetro de entrada do executável "-p" para entrar no sistema já com um projeto em aberto
- ✓ Adicionado ao cabeçalho da TRIGGER gerada a versão do ii2 que foi utilizado para a sua geração
- ✓ Duplo click sobre o painel de diagnóstico no título ORIGEM e DESTINO copia o comando SQL utilizado para a área de transferência após o diagnóstico ter sido rodado
- ✓ Criado cadastro de servidores para facilitar a troca de conexão com os bancos de dados quando se estiver testando projetos em bases de homologação e produção. Para selecionar os servidores, deve-se pressionar o botão direito do mouse na tela de conexão

VERSÃO: 3.17 (8/10/2011)

- ✓ Ajustada função CAST para identificar o tipo exato do campo, não considerando a comparação com tipos de dados similares
- ✓ Ajustada a declaração de variáveis referente a OBJETOS avulsos quando há o retorno de mais de uma variável
- ✓ Ajustada a rotina de UPDATE quando iguala os valores dos campos nas variáveis onde estava duplicando os campos chave em projetos que não eram utilizados a clausula WHERE customizada (ORACLE)
- ✓ Ajustada a gravação de campos longos (OBJETOS avulsos) no arquivo de configuração que estava se perdendo e truncando os valores inseridos
- ✓ Ajustado os valores destino da Clausula INSERT quando utilizado o comando TO_DATE em conexões ORACLE
- ✓ Inserido validação de campos de configuração de conexão antes de conectar-se aos bancos de ORIGEM e DESTINO

VERSÃO: 3.18 (22/10/2011)

- ✓ Ajustado tratamento de campos NULOS e eliminação de espaços em branco na CLAUSULA WHERE quando o banco de dados origem for ORACLE

VERSÃO: 3.19 (20/11/2011)

- ✓ Ajustado a geração do comando INSERT no ORACLE para fazer o tratamento de eliminar espaços em branco no momento da replicação
- ✓ Na aba de monitoramento, criado botões para habilitar e desabilitar os OBJETOs já criados no banco de dados

Anotações:

VERSÃO: 3.20 (30/11/2011)

- ✓ Disponibilizado opção de geração de TRIGGER como INSTEAD OF para banco de dados MSSQL
- ✓ Criado campo para marcar uma determinada tabela com integração parametrizada como INATIVAR INTEGRAÇÃO possibilitando assim efetuar a ativação gradual de OBJETOS durante a implantação de um projeto
- ✓ Criada opção que permite na coluna de COMANDO em comandos avançados a gravação de mais de uma linha por registro, facilitando desta forma a criação de blocos de comandos não sendo mais ncessário a geração de diversas linhas com a mesma configuração
- ✓ Ajustado edição de campos em OBJETOS AVULSOS que em determinadas situações permitia que fossem digitados valores não validos para a aplicação
- ✓ Ajustado edição de campos em COMANDOS AVANÇADOS que em determinadas situações permitia que fossem digitados valores não validos para a aplicação
- ✓ Em comandos avançados criado botões para mudar a ordenação dos comandos facilitando assim a manutenção e ordenação dos comandos no momento da criação do projeto
- ✓ Em comandos avançados criado um campo para ser possível optar entre a utilização dos comandos avançados padrões do projeto ou para utilizar os comandos avançados do usuário possibilitando assim a atualização do projeto sem a perda de inform customizadas
- ✓ Criado campo para condição de INSERT customizado que ficará gravado num arquivo diferenciado (CUSTOM) possibilitando assim a atualização do arquivo principal do projeto sem a interferencia das customizações ocorridas em projetos customizados
- ✓ Criado campo para condição de UPDATE customizado que ficará gravado num arquivo diferenciado (CUSTOM) possibilitando assim a atualização do arquivo principal do projeto sem a interferencia das customizações ocorridas em projetos customizados
- ✓ Criado campo para condição de DELETE customizado que ficará gravado num arquivo diferenciado (CUSTOM) possibilitando assim a atualização do arquivo principal do projeto sem a interferencia das customizações ocorridas em projetos customizados
- ✓ Criado campo de comentário customizado para tabelas que ficará gravado num arquivo diferenciado (CUSTOM) possibilitando assim que os registros de comentários efetuados pelo usuário não sejam sobreescritos durante a atualização do projeto
- ✓ Criado campo de documentação do projeto customizado que ficará gravado num arquivo diferenciado (CUSTOM) possibilitando assim que a documentação registrada pelo usuário não sejam sobreescrita durante a atualização do projeto
- ✓ Ajustado definição de layout para efetuar AUTO AJUSTE dos GRIDs dependendo da resolução do monitor que estiver sendo utilizado
- ✓ Alterado ICONE da aplicação para o padrão visual utilizado na aplicação

VERSÃO: 3.21 (17/12/2011)

- ✓ Inserido novas mensagens durante o processo de geração do OBJETO do banco para facilitar o acompanhamento e entendimento do usuário final
- ✓ Alterado mecanismo de pesquisa da estrutura das tabelas no banco de dados (MSSQL) tornando-o mais preciso

VERSÃO: 3.22 (13/1/2012)

- ✓ Ajustado busca de campos das tabelas no MSSQL que quando possui FOREING KEY estava duplicando os campos e gerando erro na geração das TRIGGERs

VERSÃO: 3.23 (1/2/2012)

- ✓ Adicionado ao StatusBar da aplicação o número da linha e coluna de navegação das caixas de texto da TRIGGER e dos OBJETOS avulsos para facilitar a consulta e conferência dos códigos gerados
- ✓ Criada mensagem resumo de registros processados após o término do processo de forçar a atualização de dados
- ✓ Melhorada mensagem apresentada quando ocorre algum erro durante o processo de forçar atualização de dados possibilitando a não apresentação continua de mensagens

Anotações:

- ✓ Ajustada falha na geração da TRIGGER (MSSQL) que estava deixando um CURSOR aberto quando uma tabela era configurada para replicar apenas as alterações e não replicar as inserções de dados
- ✓ Alterado mecanismo de pesquisa da estrutura das tabelas no banco de dados (MSSQL) na busca da chave primária (PRIMARY KEY)
- ✓ Ajustado layout de tela para melhor apresentação dos campos durante a alteração de tamanho da tela
- ✓ Inserido tecla de atalho para acesso rápido na navegação entre as ABAs de parametrização do projeto e entre as ABAs dos objetos. Também inserido atalho de acesso rápido para conexão, desconexão, seleção de projeto e filtragem de tabelas selecionadas
- ✓ Acrescentada validação de chave primária no momento em que se força a atualização dos dados de uma tabela
- ✓ Ajustada identação dos comandos avançados quando inserida mais de uma linha de código programável
- ✓ Ajustado a montagem da TRIGGER quando marcado novos campos para ativação da Interface que antes era necessário mudar de tabela para que a alteração fosse reconhecida
- ✓ Ajustado FOCO de objeto quando se efetuava a desconexão do banco de dados em algumas versões do Windows
- ✓ Ajustado ERRO de comando ao se buscar estrutura de dados (MSSQL)
- ✓ Adicionado mensagem de segurança para não perder as configurações de INS, UPD e DEL durante a manipulação de tabelas já parametrizadas, sendo necessária a confirmação do usuário no manuseio das mesmas

VERSÃO: 3.24 (10/2/2012)

- ✓ Alterado mecanismo de pesquisa da estrutura das tabelas no banco de dados (MSSQL) na busca da chave estrangeira (FOREIGN KEY)
- ✓ Adicionado contador de tempo e previsão de tempo restante para término da rotina de forçar atualização de dados

VERSÃO: 3.25 (20/2/2012)

- ✓ Homologado sistema Senior para validação da licença na versão ONIS
- ✓ Homologado sistema Escalamax para validação da licença na versão ONIS

VERSÃO: 3.26 (4/3/2012)

- ✓ Atualização de validação do arquivo de licenciamento

VERSÃO: 3.27 (30/3/2012)

- ✓ Atualização de validação do arquivo de licenciamento

VERSÃO: 3.28 (1/4/2012)

- ✓ Ajuste na leitura e validação do arquivo de licenciamento

VERSÃO: 3.29 (1/5/2012)

- ✓ Habilitada rotina de DE/PARA para manipulação de dados com arquivos tipo DBF
- ✓ Adicionado visualizador de dados para abrir tabelas .DBF com os valores respectivos de cada campo
- ✓ Opção de alteração de PK de arquivo DBF para evitar duplicidade de dados na geração do arquivo de saída na opção de Forçar Carga de Dados
- ✓ Melhorado rotina de geração de arquivos .TXT
- ✓ Melhorado rotina de manipulação de arquivos .DBF
- ✓ Adicionado funcionalidade de exportação e importação de parametrização de tabelas de um projeto para arquivo .EXP pressionando-se botão direito do mouse sobre a tabela

Anotações:

VERSÃO: 3.30 (14/5/2012)

- ✓ Acertado problema com aplicação de TRIGGER no banco, quando trabalhado com projetos em ORACLE
- ✓ Ajuste na validação da proprietária da ferramenta

VERSÃO: 3.31 (1/6/2012)

- ✓ Ajustado a montagem da CLAUSULA WHERE quando um dos campos chaves era tratado através de valor FIXO para integrações de banco de dados ORACLE que não estava considerando como STRING
- ✓ Na visualização do OBJETO gerado, foi implementado a opção de destaque para palavras reservadas
- ✓ Na opção de cadastramento de objetos avulsos, foi implementado a opção de destaque para palavras reservadas
- ✓ Na opção de edição dos comandos avançados, foi implementado a opção de destaque para palavras reservadas
- ✓ Criado opção para gerar PROCEDURE no lugar de TRIGGER como objeto de integração de dados (apenas ORACLE)
- ✓ Criado opção de filtro de tabelas no momento da conexão para facilitar a pesquisa das mesmas (apenas ORACLE)
- ✓ Adicionado editor para facilitar o cadastro de valores em campos destino que não possuem seus respectivos na origem
- ✓ Alterado tipo de fonte na tela de comandos avançados para facilitar a identação dos códigos programáveis
- ✓ Ajustado problema que ocorria no momento da leitura dos arquivos .INI quando eram abertos 2 projetos com comentários cadastrados
- ✓ Ajustada funcionalidade de integração entre bancos de dados distintos (ORACLE -> SQL)
- ✓ Liberada versão para trabalhos com SYNONYMS no ORACLE

VERSÃO: 3.32 (20/6/2012)

- ✓ Atualização da validação da proprietária do sistema

VERSÃO: 3.33 (22/8/2012)

- ✓ Ajustada funcionalidade de integração entre banco DBF e Oracle

VERSÃO: 3.34 (17/12/2012)

- ✓ Ajustada gravação da documentação referente aos objetos avulsos. Havia um problema que estava gravando apenas a primeira linha do documentário
- ✓ Alterada chamada de DLL de conexão com banco de dados DBF, para não impactar projetos que utilizem apenas bancos de dados MSSQL e ORACLE
- ✓ Ajustado gravação de campos destino no INSERT e UPDATE quando os mesmos são alterados via botão de CAMPOS X VALOR
- ✓ Adicionada opção que possibilita escolher o campo que será utilizado para UPDATE no momento que a opção de forçar a carga de dados for selecionada
- ✓ No MSSQL acertado a declaração de variáveis de campos quando o tipo do campo no banco for definido como MAX. O mesmo não estava sendo tratado corretamente gerando erro na aplicação da TRIGGER no banco de dados
- ✓ Ao montar a integração de tabelas e selecionar os campos que farão parte da integração, o sistema traz campos com o mesmo nome entre as tabelas automaticamente como sugestão.
- ✓ Inserido comando SET IDENTIFY ON ao utilizar a opção de forçar a carga de dados em bancos de dados do tipo MSSQL

VERSÃO: 3.35 (30/3/2013)

Anotações:

- ✓ Ajustado apresentação do LENGTH dos campos tipo DATE, DATETIME, TIME e IMAGE para apresentação ao usuário na tela de ligação de campos (MSSQL)
- ✓ Adicionada opção para listar os databases existentes no banco de dados na tela de conexão (MSSQL)
- ✓ Ajustada funcionalidade de filtrar apenas as tabelas selecionadas que não estava retornando todas as tabelas quando o checkbox era desmarcado
- ✓ Ajustada funcionalidade para habilitar e desabilitar TRIGGERs do projeto selecionado no banco de dados (MSSQL)

VERSÃO: 3.36 (16/8/2013)

- ✓ Criada opção de tirar os espaços dos campos na CLAUSURA WHERE quando montado os campos de UPDATE
- ✓ Criada opção de visualização dos dados das tabelas de ORIGEM e DESTINO para facilitar a analise dos dados que serão integrados
- ✓ Criado conceito de Tabela Transitório (em DESTINO) para integração de tabelas que só devem receber INSERT para serem tratadas posteriormente por uma rotina de migração de dados pelo sistema DESTINO
- ✓ Disponibilizada opção de seleção de esquema repositório de integração, onde o processo de integração é executado em duas etapas, passando por uma tabela NATIVA do II2 no meio do processo para posterior encaminhamento para o banco de dados DESTINO
- ✓ Alterado o esquema de busca de lista de tabelas no ORACLE que quando não tiver acesso para listar da USER_TABLES, fará a busca através da ALL_TABLES
- ✓ Alterado o esquema de busca de lista de colunas da tabela selecionada no ORACLE que fazia acesso a USER_TAB_COLUMNS passando a acessar as informações em ALL_TAB_COLUMNS
- ✓ Alterado layout de ABA de Monitoramento e Configuração do banco de dados para melhor experiência de uso dos usuários
- ✓ Disponibilizada opção de alteração visual da ferramenta com estilo personalizado

Anotações: